AD-A053 554 NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA F/G 17/2
CHANNEL ADAPTIVE RECEIVER TEST-BED SOFTWARE-CONTROLLED HARDWARE--ETC(U)
JAN 78 M D JUNIPER

UNCLASSIFIED NOSC/TD-141 NL

| OF |
AD
A053554

END
DATE
FILMED
6-78
DDC

NOSC

12

NOSC TD 141

Technical Document 141

CHANNEL ADAPTIVE RECEIVER TEST-BED

Software-controlled hardware for studying underwater acoustic communication receivers uses an adaptive filter and can emulate other adaptive systems

MD Juniper

19 January 1978

NAVAL OCEAN SYSTEMS CENTER
SAN DIEGO, CALIFORNIA 92152

NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA 92152

AN ACTIVITY OF THE NAVAL MATERIAL COMMAND

RR GAVAZZI, CAPT, USN
Commander

HL BLOOD
Technical Director

ADMINISTRATIVE INFORMATION

The work reported herein was sponsored by NAVELEX 310/ NAVSEA 06H1. It was performed under the Acoustic Communications Exploratory Development block program, program element 62721N, project F21221, task area XF21221091, NOSC work unit CM57.

Released by
RH Hearn, Head
Electronics Division

Under authority of
DA Kunz, Head
Fleet Engineering Department

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(14) NOSC/TD-141

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NOSC Technical Document 141 (TD 141)
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): (6) CHANNEL ADAPTIVE RECEIVER TEST-BED
Software-controlled hardware for studying underwater acoustic communication receivers uses an adaptive filter and can emulate other adaptive systems.

5. TYPE OF REPORT & PERIOD COVERED: (9) Interim rept.,
6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): (10) MD Juniper
(16) F21221
(17) XF21221091

8. CONTRACT OR GRANT NUMBER(s): (12) 35 p.

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Naval Ocean Systems Center
San Diego CA 92152

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
62721N, F21221, XF21221091, (NOSC) CM57

11. CONTROLLING OFFICE NAME AND ADDRESS:
Naval Electronic Systems Command (ELEX 310) and
Naval Sea Systems Command (NSEA 06H1)
Washington DC 20360

(11) 12. REPORT DATE: 19 Jan [redacted] 978
13. NUMBER OF PAGES: 32

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):

15. SECURITY CLASS. (of this report): Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Adaptive communications	Filter – adaptive
Adaptive systems	Input output devices
Digital filters	Minicomputers
Executive control programs	Underwater communications

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document describes a software-controlled experimental test-bed designed primarily to implement a receiver for underwater acoustic communications. It is also useful in evaluating the feasibility of incorporating adaptive filtering in various other signal processing systems. A least-mean-square filter provides the major source of processing; a minicomputer and peripheral devices are used for the transfer, storage, input, and output of data. Concurrent Pascal is used to implement a flexible user-specified real-time operating system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102 LF 014 6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

393 159

OBJECTIVE

Develop a software-controlled experimental test-bed designed primarily to implement a receiver for underwater acoustic communications. Include in the design the necessary generalizations to make the system useful in evaluating the feasibility of incorporating adaptive filtering in various other signal processing systems and as an experimental laboratory for various adaptive techniques.

RESULTS

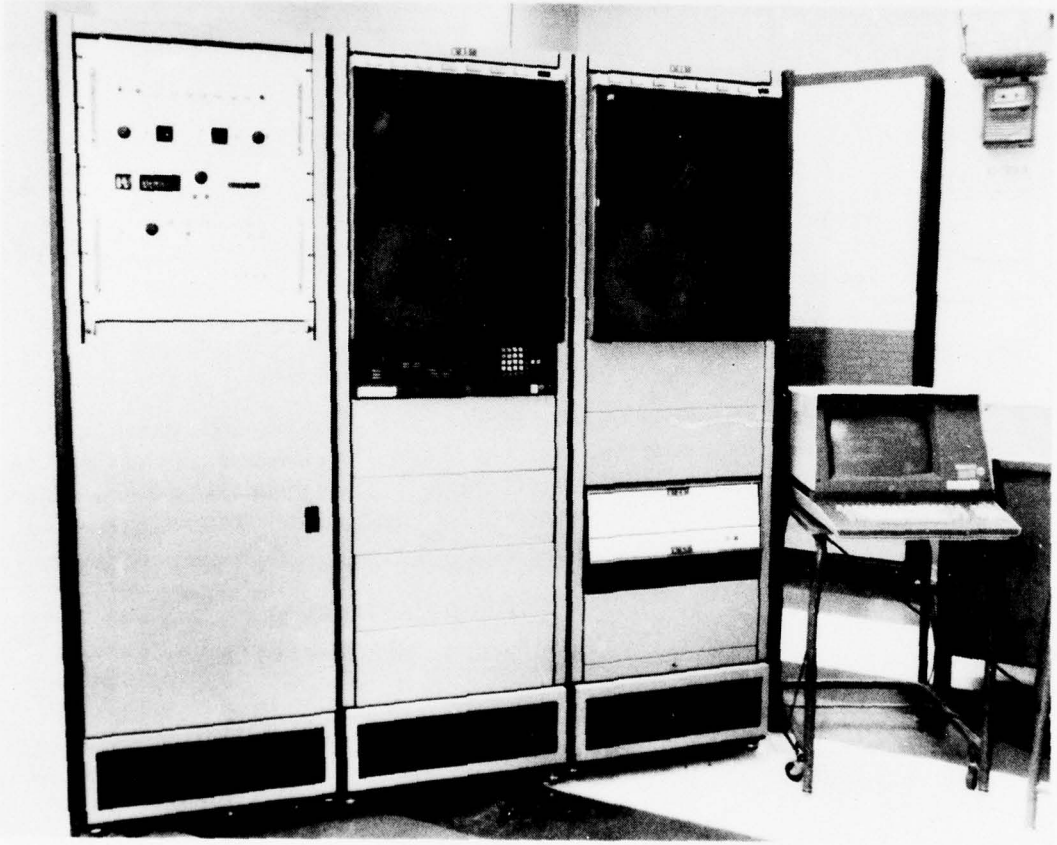
Special-purpose hardware was developed. It is controlled by a minicomputer that runs under a concurrent operating system. All filter parameters as well as the actual filter configurations are determined by control data from the minicomputer. Work is currently underway both to determine the receiver operational characteristic curves of the system and to process actual sea data.

CONTENTS

1 INTRODUCTION

Although the test-bed described within this document was designed as a means of evaluating signal enhancement through the incorporation of adaptive filters in underwater acoustic communication system receivers, it can be used in implementing and evaluating a variety of systems. The design includes the necessary generalizations to make the system useful as an experimental laboratory for various adaptive techniques. Figure 1 is a view of the test-bed.

Special-purpose hardware is controlled by a minicomputer that runs under a concurrent operating system. All filter parameters as well as the actual filter configurations are determined by control data from the minicomputer. All the system hardware is described, with special emphasis on the NOSC-designed and -built hardware. The NOSC-designed operating system is also briefly mentioned. An attempt is made to show both the capabilities and limitations of this system.



LRO 7303-12-77B

Figure 1. Test-bed hardware.

2 SYSTEM DESCRIPTION

This system, equipped with the flexible operating system described in section 8, may be used to perform many diverse functions. This section relates the major hardware

components and their interconnecting data paths. Figure 2 shows the major test-bed interconnections.

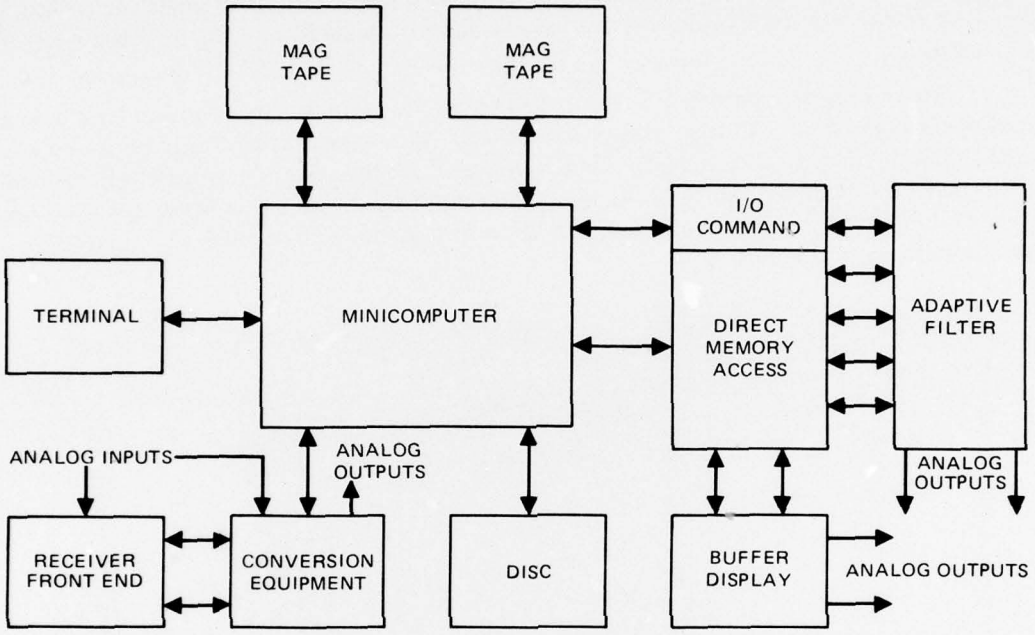


Figure 2. Test-bed component interconnections.

The specific type of hardware used for each functional component is shown in the table below. The procured hardware is described in greater detail in section 3, and the NOSC-built components are separately described in sections 4–7. The concurrent operating system device drivers, which are already written (section 8), automatically take care of many of the details required to operate the equipment.

SPECIFIED TEST-BED HARDWARE

FUNCTION	SPECIFIC TYPE
Procured hardware	
Minicomputer	Interdata 7/16
Terminal	Tektronix 4023
Mag tape	Wanco Mod 11
Disc	Diablo Series 40
Conversion equipment	Computer Products Inc RTP interface
NOSC-built hardware	
Adaptive filter	(See section 4)
Direct memory access	(See section 5)
*Receiver front end	(See section 6)
Core buffer display	(See section 7)

*Resident in the conversion equipment chassis.

The following examples constitute systems and functions that can be emulated by this adaptive filter test-bed. There are of course many other more general ways the hardware may be configured.

ALTERNATIVE FUNCTIONS

Data from the minicomputer can set the processor to implement any of four filter functional configurations:

Correlator

Single-input adaptive filter

Multiple-input filter

Slaved filter

Multiple filters using any combination of these configurations are also possible. The adaptive filter hardware can be configured by software into multiple filters with the following constraints:

Total of 2048 complex or 4096 real weights

Total of 8192 complex or 16384 real samples in the input vectors

Total of 16 independent filters

Total of 32 input channels

All real or all complex filters

For combinations exceeding these limits, the hardware may be time-shared from the minicomputer. In this way, for example, a filter with more than 32 input channels may be programmed.

Analog operation without the minicomputer is available with a maximum of one real desired response channel with seven input channels.

SYSTEM APPLICATIONS

The following are typical system applications made possible by the adaptive filter:

Single-reference noise canceller
- 1 input
- 500 complex weights
- Complete update cycle, 64 μs
- Maximum input frequency, 7.8 kHz

Multiple-reference noise canceller
- 8 inputs
- 500 real weights each input channel
- Complete update cycle, 512 μs
- Maximum input frequency, 975 Hz

Adaptive beam former
- 32 input channels
- 125 real weights each input channel
- Complete update cycle, 512 μs
- Maximum input frequency, 975 Hz

Adaptive channel modeling
- 1 input
- 2000 complex weights
- Complete update cycle, 250 μs
- For a maximum input frequency of 100 Hz, (125-Hz sample frequency for complex operation), the filter will run 32 times faster than real time.

Line enhancer
- 1 input
- 250 complex weights
- Complete update cycle, 32 μs
- For a maximum input frequency of 500 Hz with a sample rate only slightly higher, this filter will run about 60 times real time.

Fixed replica correlator
- Filter length, 500 real samples
- Complete update cycle, 64 μs
- Maximum input frequency, 7.8 kHz
- For a 250-Hz bandwidth (500-Hz sample rate), the 500 samples will hold a 1-second period of input data. This correlator could run more than 30 times real time.

Toggled reference correlator
- Filter length, 500 real samples
- Complete update cycle, 64 μs
- Full correlation cycle, 32 ms
- This correlator could run about 31 times real time, or equivalently, could run 30 input channels in real time. The present driver computer configuration does not have the data buffer capacity to handle this many channels.

Other possible adaptive systems include channel equalizers and adaptive predictors.

3 PROCURED HARDWARE

The procured hardware consists of the minicomputer and four peripheral devices. The important parameters (those which indicate major capabilities) of each item are listed. For additional information on hardware items, consult the manufacturers' user manuals listed under REFERENCES.

MINICOMPUTER

A thorough study of available minicomputers was made before purchasing the Interdata 7/16. This study and the reasons for choosing this particular minicomputer are documented in NOSC technical note NUC TN 1408.[1]

The Interdata 7/16 is a general-purpose 16-bit minicomputer. It has 64k bytes of directly addressable memory, 16 general registers, 104 instructions, hardware interrupt vectoring for up to 255 devices, and four DMA channels that run up to 2M bytes per second.

The memory has a 750-ns cycle time with memory parity for data and instruction protection. Another feature is programmable memory protect, which permits or disables writing into 1k-byte blocks of memory under software control. All 16 general registers are available to the user for any purpose, allowing a convenient storage place for partial results, constants, base addresses, or anything else.

Some of the included options are power fail detection/auto restart, hexadecimal display front panel, high-speed arithmetic logic unit (ALU), universal clock, and hardware floating point multiply and divide.[2,3] The Interdata 7/16 parameters are as follows:

16-bit general-purpose minicomputer
64k-byte memory with parity, 750-ns cycle time
16 general registers
104 instructions
Interrupt vectoring for 255 devices
Four DMA channels, 2M bytes per second
Memory protect (programmable)
Power fail detection/auto restart
Hexadecimal front panel display
High-speed ALU
Universal clock
Hardware floating point/multiply and divide

COMPUTER DISPLAY TERMINAL

The chosen computer terminal is the Tektronix 4023. It provides an operator control/interface for the system and returns required system information to the operator. The Tektronix 4023 parameters[4] are as follows:

Half or full duplex
Scrolling display, 24 lines by 80 characters
Selective baud rate, 8 positions 110–9600 baud
Local/on-line and direct/buffer modes
Display editing and special formatting
94 ASCII characters printed using 5 by 7 dot matrix

---

[1]NOSC Technical Note NUC TN 1408, A Minicomputer Study, by CC Ross, August 1974. NOSC TNs are informal documents intended chiefly for internal use
[2]Interdata Inc User's Manual, Pub 29-261R03, July 1974
[3]Interdata Inc 16 Bit Series Reference Manual, Pub 29-398R01, August 1974
[4]Tektronix Inc 4023 Computer Display Terminal Users Instruction Manual, September 1974

MAGNETIC TAPE TRANSPORT

The Wanco Mod 11 digital magnetic tape transport is used mainly to load programs into and out from the system and to load and save digitized data. The major parameters[5] are as follows:

Nine tracks
800 characters per inch
75 inches per second
60 000 characters per second
Remote or local control
Maximum of 10.5-inch diameter tape reel

DISC DRIVE

The Diablo Series 40 disc drive is used to store data files, copies of the operating system, and other desired information. The disc drive consists of a removable head system with two discs, one a removable disc cartridge and one fixed. Each disc holds 5M bytes of data. The major parameters[6,7] are as follows:

Fixed and removable disc
Total 10M-byte capacity
1500 rpm
20-ms average latency
Maximum data transfer 180k bytes per second
Block sizes 256 to 12 288 bytes
Front panel write protect
Power loss protection

CONVERSION EQUIPMENT

The Computer Products Inc RTP real-time processing interface contains several conversion equipment. The main components of the conversion equipment are one 16-input A/D converter, three D/A converters, a conversion equipment chassis, and a controller. This peripheral device permits the system to have real-time analog computer control of these inputs and outputs. The parameters for the components of the conversion equipment are as follows:[7-10]

A/D converter
- 16 single-ended inputs
- Input limits ±2.56 V
- 12-bit converter sign extended to 16 bits
- Maximum sample rate 10 kHz
- Accuracy ± 0.05% of full scale ± 1/2 LSB

[5]Wanco Inc Magnetic Tape Transport Operation and Maintenance Manual, 101100, June 1973
[6]Diablo Systems Inc Series 40 Disk Drive Maintenance Manual, April 1972
[7]Computer Products Inc 8-Channel and 16-Channel High-Level Analog Input Modules, PM021-001K, January 1974
[8]Computer Products Inc Analog Output Modules, PM021-002K, August 1974
[9]Computer Products Inc Digital Input/Output and Analog Output Sections Instruction Equipment PM070-004D, October 1974
[10]Computer Products Inc Conversion Equipment Controller Programming Specification, 02-2980-001, July 1974

D/A converter
- Input 12 bits binary
- Output accuracy ± 0.05%
- Settling time 50 μs
- Slew rate 0.4 V/μs
- Output ±5 V

Conversion equipment chassis
- Accepts up to 16 input/output modules
- Module address determined by slot location

Controller
- Two 16-bit by 64-word fifos (first-in, first-out memories), with one for input and one for output
- Programmed interrupt-driven control
- Reduced interrupt-service overhead

4 ADAPTIVE FILTER

The adaptive filter (see fig 3) is a least-mean-square (LMS) hardware processor designed to allow very fast implementation of adaptive filters by using the LMS algorithm for updating the weights (see fig 4). Familiarity with this algorithm is assumed in the following discussions. (For additional information on this algorithm consult references 11–16.) The hardware is controlled from a minicomputer, with data available directly from the computer memory through four direct memory access (DMA) channels (fig 5). This section contains the data necessary to understand the capabilities, the operating characteristics, and the minicomputer interfacing for the filter. For information on controlling the DMA channels, see section 5.

---

[11] Stanford University Stanford Electronics Laboratory Report SU-SEL-66-126, Adaptive Filters 1: Fundamentals, by B Widrow, December 1966

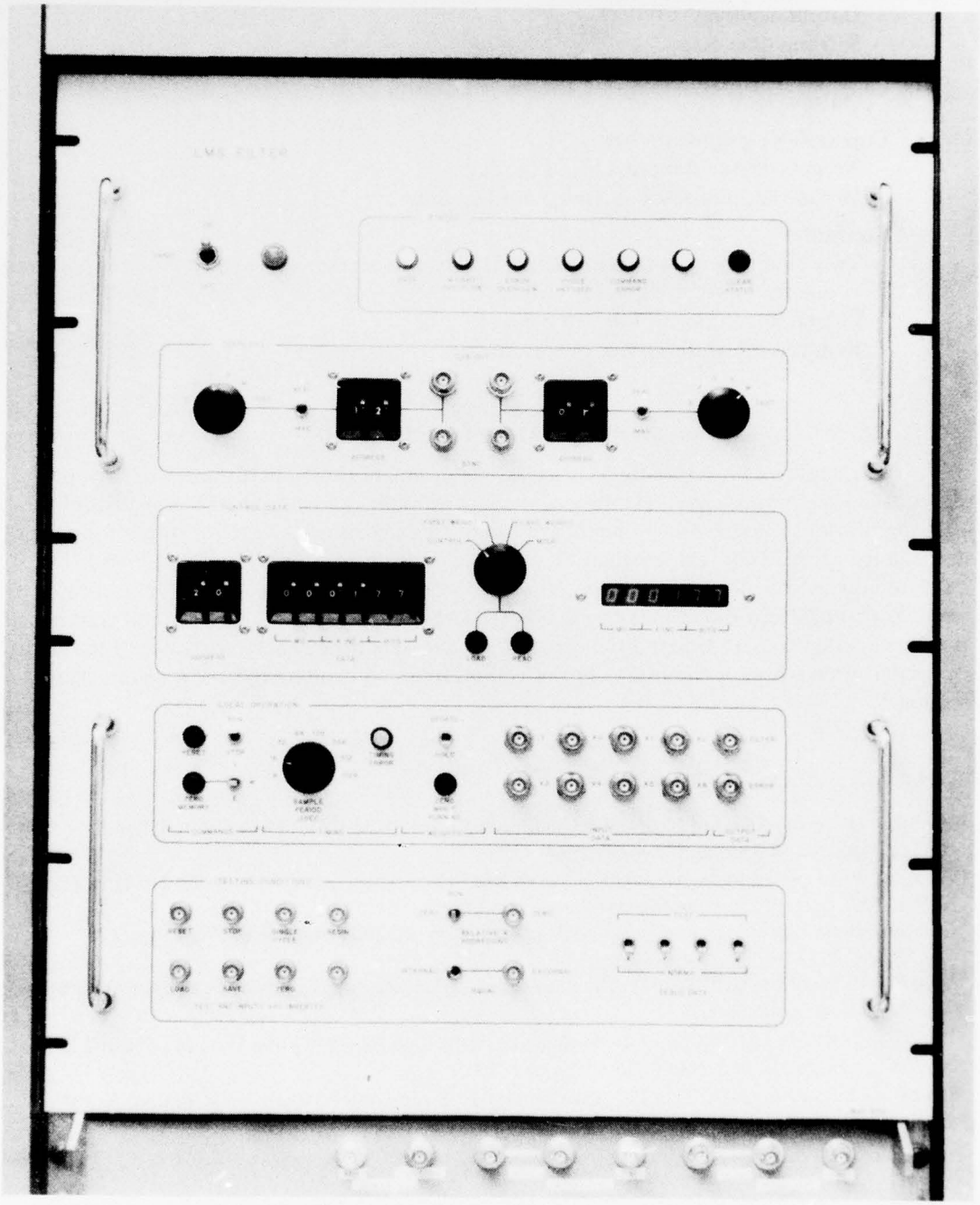
[12] Widrow, B, Adaptive Filters, Aspects of Network and System Theory, R Kalman and N DeClaris, ed, p 563–587, Holt, Rinehart, and Wilson, New York, 1971

[13] Adaptive Noise Cancelling: Principles and Applications, Proceedings of the IEEE, vol 63, no 12, December 1975

[14] NOSC Technical Note NUC TN 837, The Basic Principles of Adaptive Systems with Various Applications, by JM McCool, September 1972.

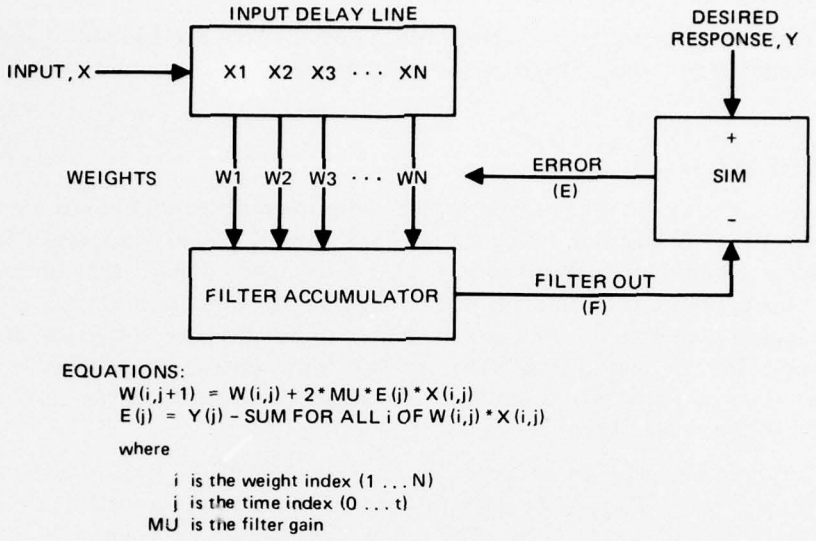
[15] NOSC Technical Note NUC TN 1476, An Analysis of the LMS Adaptive Filter Used as a Spectral Line Enhancer, JR Zeidler and DM Chabries, February 1975.

[16] NOSC Technical Note NUC TN 1437, The Complex LMS Algorithm, B Widrow, JM McCool, and MS Ball, October 1974.



LRO 7307-12-77B

Figure 3. Adaptive filter front panel.



EQUATIONS:

$W(i,j+1) = W(i,j) + 2 * MU * E(j) * X(i,j)$

$E(j) = Y(j) - \text{SUM FOR ALL } i \text{ OF } W(i,j) * X(i,j)$

where

i is the weight index (1 . . . N)
j is the time index (0 . . . t)
MU is the filter gain

Figure 4. Adaptive transversal filter.

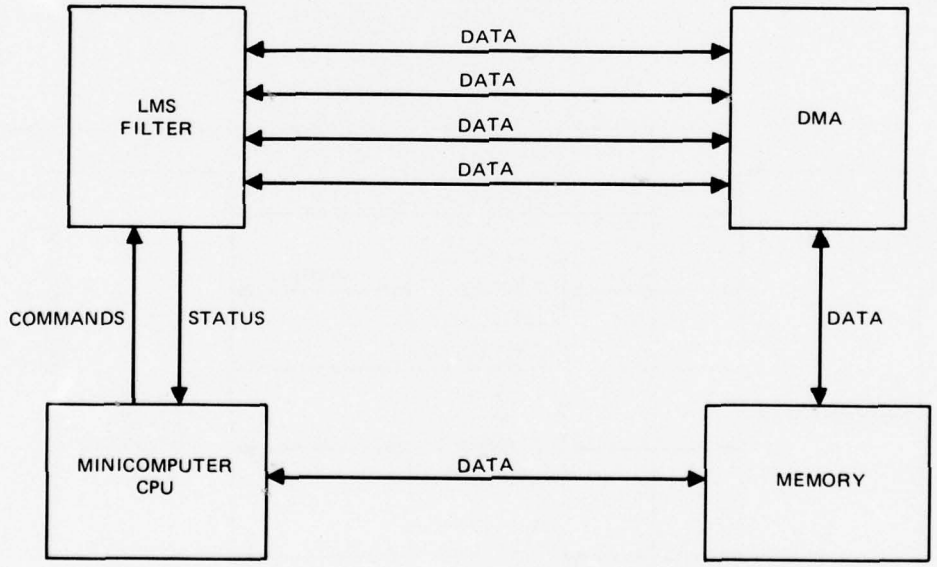


Figure 5. LMS filter connections to minicomputer.

FUNCTIONAL DESCRIPTION

The following paragraphs describe the capabilities of the LMS filter and its more detailed parameters in terms of maximum sizes and ranges.

CAPABILITIES

The LMS filter can be programmed to have many filter configurations or to do simple cross correlation. In addition to the simple LMS filter, the main configurations are multiple input filters, slave filters, multiple independent filters, and combinations of these. It is designed to use complex arithmetic, so real arithmetic is available as a subset. Every filter in any configuration may have a unique mu (gain) and number of weights, and in each case these weights may be spaced at different intervals along the X, or input, vector (ie, weights are applied only to every nth X sample). The weights for each filter can also be frozen independently of the other filters.

It is possible to preset or copy the complete state of the LMS filter from the minicomputer (see fig 6). This will be done only when all filters in the current configuration have completed a full update cycle. The state of all filters is determined by the configuration control words (which cannot be copied back into the minicomputer) and by the contents of the X, W, and ERROR memories. All or any subset of the X and W memories may be loaded from, or copied into, the minicomputer; all of the ERROR memory is always loaded or copied. This allows the user to initialize filter configurations to any desired state. Furthermore, one configuration can be time-shared with others by saving the state of the machine and later restoring it.

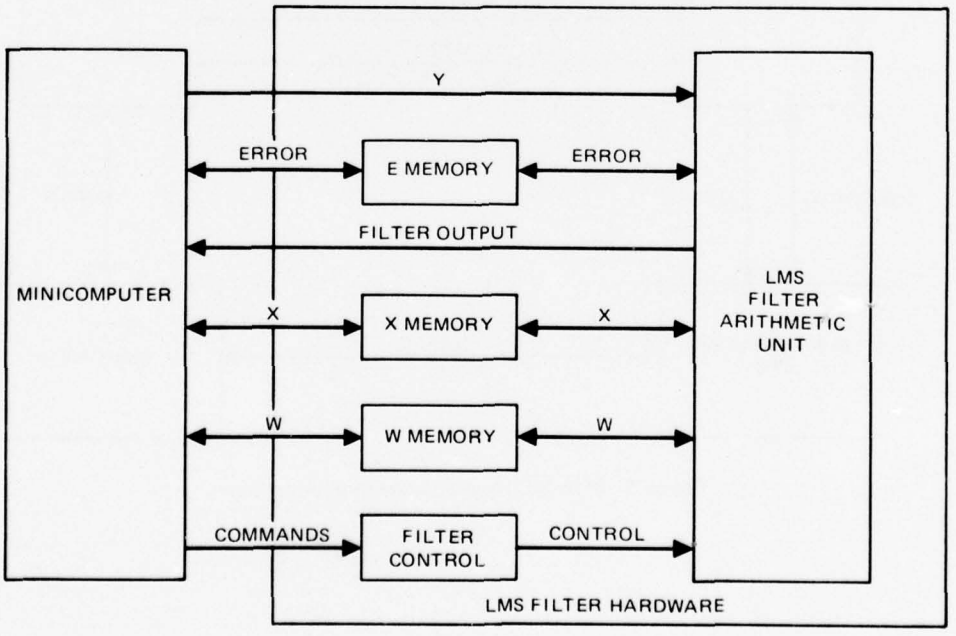


Figure 6. Data paths for generalized LMS filter hardware.

Local operation on real-time analog inputs is a further capability. Any real filter requiring only one Y (desired response) input and up to seven X (input vector) channels may be run from the front panel of the filter.

PARAMETERS

The important parameters and their values for the LMS filter hardware are as follows:

Arithmetic is 1's complement with maximum magnitude 1.0 minus 1/128

X, Y, error, and filter data are 8 bits

Filter accumulator has 28 bits so that sequential summation values are not lost

The weights are 32 bits

One XW product and one weight update are done every 125 nanoseconds

Weight memory is 2k (2048) words by 32 bits for complex filters, increasing to 4k (4096) words for real filters

X memory is 8k (8192) words by 8 bits for complex filters, increasing to 16k (16 384) words for real filters

Mu is adjustable from 2^{-9} to 2^{-24} in powers-of-two steps

Maximum number of independent filters in any configuration is 16

Up to 32 references are available that can be arranged with the filter in any way

Weights may be spaced at intervals of from 1 to 63 along the X input vector

If overflow of the filter output, error, or weights occurs, the magnitude is limited to ±(1.0 minus 1/128)

PROGRAMMING

The LMS filter is controlled and configured entirely by minicomputer software. Commands to execute the various hardware functions are given by direct I/O instructions from the computer. The parameters which determine the filter configuration are contained in a table of control data words transferred to the filter hardware by the minicomputer.

When the filter is running, it requires data for the input signal and for the desired response; output data are available from the filter output and from the error signal. All data except direct I/O instructions are transferred through four independent direct memory access (DMA) channels dedicated to the filter operation: one for X and the weights, one for Y, one for the filter output and control data, and one for the error signal. The following paragraphs describe how the LMS filter is programmed. The data and control paths are illustrated in figure 7.

The following conventions are used in all subsequent discussions:

X = filter input
Y = desired response input
F = filter output
E = error output
W = weights
Bit 0 = least significant bit of all data words

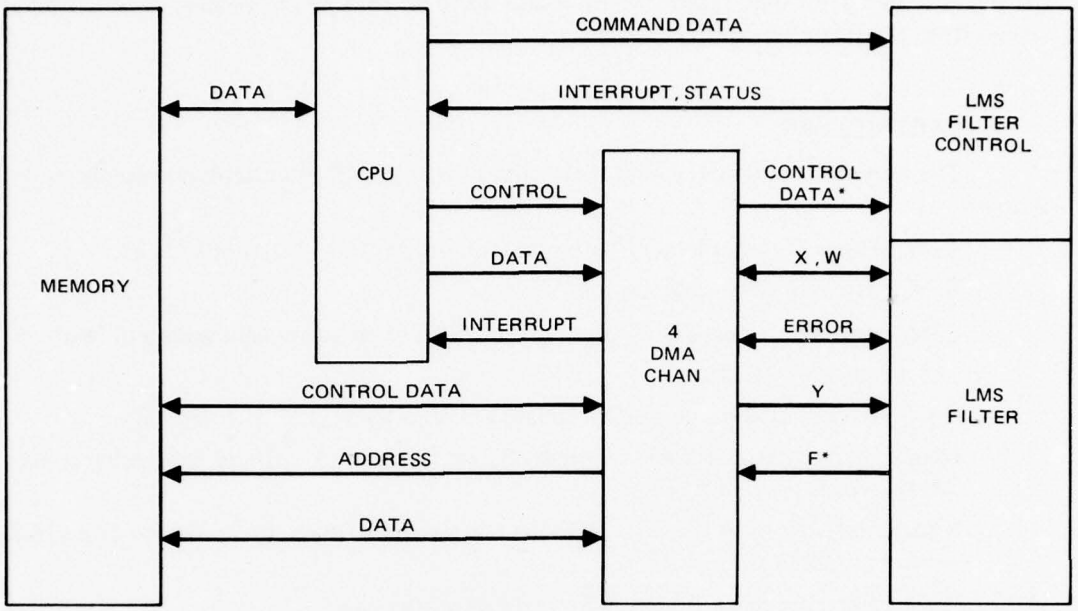


Figure 7. Data and control paths for LMS filter.

COMMANDS

Direct commands to the LMS filter hardware are used to start and stop the filter and to load or save the data in the filter memories. Commands must be issued one at a time by the minicomputer CPU. When the filter has completed a command, it sends an interrupt to the minicomputer. If a command is issued before the previous one has been completed, it will be ignored and a command error interrupt will occur.

A SINGLE CYCLE command will run one update cycle of each filter in the current configuration as determined by the control data words. This is called one configuration cycle. A START command starts running consecutive configuration cycles until data are exhausted or until a STOP command is received. A STOP command always allows the currently executing configuration cycle to finish.

The LOAD and SAVE commands for the X or W memory require special consideration. First, because of addressing and buffering considerations, the mode word must already have been loaded into the filter. Second, these commands require the user to consider limits on the amount of data to be transferred.

In most cases only a small subset of the total memory needs to be loaded or saved. Therefore, a command to load or save either the X or W memory will first read from the filter DMA channel an initial address for the X or W memory, then will transfer the data through the appropriate DMA channel until the channel indicates a refusal to transfer more data (cycle refused term). This terminates the LOAD or SAVE command.

In choosing the X memory limits, the user must know that all X values are packed sequentially in the X memory starting at location O. The addresses in the X memory are

relative to a hardware base register which is incremented upon completion of each configuration cycle. The number of X locations used is dependent upon the weight spacing.

An 8-bit data word is used to command the filter. The bit assignments are listed in the following table. Note that the command bits 5, 6, and 7 are unused. Also, the commands for ZERO CONTROL and SAVE CONTROL are ignored except that an interrupt is given to acknowledge that the command was received.

COMMAND DATA BIT DESCRIPTION

COMMAND BITS 2	1	0	COMMAND FUNCTION
0	0	0	RESET
0	0	1	STOP
0	1	0	SINGLE CYCLE
0	1	1	START
1	0	0	LOAD
1	0	1	SAVE
1	1	0	ZERO
1	1	1	INIT

COMMAND BITS 4	3	MEMORY CODE
0	0	CONTROL
0	1	X
1	0	WEIGHT
1	1	ERROR

COMMAND BITS 7	6	5	
			UNUSED

Here is a brief description of each command:

RESET — Clear the entire LMS hardware (except the memories) regardless of the current state.

STOP — Stop after finishing the current update cycle on all filters in the current configuration.

SINGLE CYCLE — Execute or finish one update cycle on all filters in the current configuration.

START — Begin execution of a new filter cycle or resume the current operation.

LOAD — Load data into the X, W, E, or CONTROL memories from the DMA channels.

SAVE — Save the current values in the X, W, or E memory by writing into the DMA channels.

ZERO — Set all values in the X, W, or E memory to zero.

INIT — Clear the hardware as in RESET, except no interrupt is returned to the minicomputer. (Used by the operating system.)

The results of giving each of these commands while the filter is in various states is summarized under interrupt and status which follows.

If one of the DMA channels gives a cycle refused indication (runs out of data) while the filter is running, the filter hardware will stop. If this occurs on the X or Y channel when a full configuration cycle is complete, the result is the same as if a STOP command were given. An interrupt is returned, but the status bit for "unexpected cycle refused" is not set. If the hardware is in midcycle or the cycle refused is on the F or E channel, a WAIT condition results and the status is set to indicate that the filter is BUSY with an unexpected cycle refused. To continue in either case, refresh the empty DMA channel and give a START or SINGLE CYCLE command. The SINGLE CYCLE command will clear the WAIT state and finish the current filter cycle, then stop.

INTERRUPT AND STATUS

The LMS filter is designed to keep the minicomputer as informed as possible concerning the current state of the hardware. To this end, an interrupt line and eight status bits are provided. The important status bits are also displayed on the front panel. It is intended that whenever an interrupt occurs, the computer should read the status to see if an error has occurred or if the interrupt is the normal response to a command.

The status bit assignment and definitions are summarized in the table that follows. Status bits 0 through 3 were chosen to have the same meaning as the least significant bits in the minicomputer status word convention. Bits 4 and 5 are not necessarily error indicators, but are simply informative. Bit 6 is normally an error, but the filter enters the WAIT state if the filter was running and in midcycle at the time this cycle refused occurred. A START command after the appropriate DMA buffer is refreshed will cause the filter to resume operation. Status bit 7 indicates an error, but this error did not interfere with the operation going on in the filter at the time. When the minicomputer reads the filter status, bits 4 through 7 are set to zero.

STATUS BIT ASSIGNMENT AND DESCRIPTION

BIT	MEANING	CONDITIONS
0	Device unavailable	Front panel is enabled so the computer interface is disabled
1	End-of-medium	Always = 0 (not used)
2	Check status	At least one of the status bits 3–7 is set
3	Device busy	Filter is running, in the wait state, or in the middle of a LOAD, SAVE, or ZERO
4	Weight overflow	The magnitudes of either the real or imaginary weights have been limited since the status was last checked
5	Error overflow	Same as bit 4 for the error terms
6	Unexpected cycle refused	Unexpected cycle refused occurred since the status was last checked
7	Command error	A command error occurred since the status was last checked

Except for INIT, an interrupt is always given when a command has been received and completed. In some cases, such as START, the interrupt is given immediately. At other times, such as LOAD, the interrupt is not returned until the specified operation is complete. In all cases where an error has occurred, an immediate interrupt will be sent to the minicomputer. A summary of the commands and the interrupt response is contained in the following table.

INTERRUPT RESPONSE TO THE COMMANDS

COMMAND	WHEN VALID	RESPONSE* WHEN BUSY	RESPONSE* WHEN NOT BUSY
RESET	Always	RESET, II	RESET, II
STOP	Running or not busy	(NOTE 5), IC	NOP, II
SINGLE CYCLE	Not busy	(NOTE 2), IC	(NOTE 6), IC
START	Not busy	(NOTE 2), NONE	(NOTE 6), NONE
LOAD	Not busy	ERROR, II	LOAD MEMORY, IC
SAVE	Not busy	ERROR, II	SAVE MEMORY, IC
ZERO	Not busy	ERROR, II	ZERO MEMORY, IC
INIT	Always	RESET, NONE	RESET, NONE

*TYPES OF RESPONSE:

II = Immediate interrupt
IC = Interrupt when completed
NOP = No operation performed
NONE = No interrupt is given

NOTES:

(1) An invalid command interrupts immediately and sets status bit 7 to a logical 1. The bad command is ignored.

(2) When the filter has paused in the WAIT state, START or SINGLE CYCLE will cause the filter to resume operation. START and SINGLE CYCLE are errors when the filter is BUSY but not in the WAIT state.

(3) For LOAD, SAVE, and ZERO, the appropriate memory is determined by command bits 3 and 4.

(4) BUSY = filter running or doing a LOAD, SAVE, or ZERO.

(5) STOP is an error during a LOAD, SAVE, or ZERO.

(6) SINGLE CYCLE and START do not cause interrupts directly, but the filter does interrupt when it stops running. This is roughly equivalent to a case of "interrupt when completed."

CONTROL DATA

The LMS filter configuration and its variable filter parameters are set by using one or more control words. Each control word consists of three parts: (1) control bits with configuration and sequencing data, (2) the initial address to use in the weight memory, and (3) the final weight memory address. Each change in one or more of the variable filter parameters is set by a new control word. There may be up to 32 of these control words, each of which is a total of 48 bits in length and is stored in the control memory. These control data are transferred through the filter DMA channel from the minicomputer; control words are read until a cycle bit is encountered (word 0, bit 3 = 1). Each control word is split into three 16-bit data words as follows:

WORD 0 = control bits
WORD 1 = first W address
WORD 2 = last W address

In addition to the control words, a MODE word is sent at the beginning of this buffer to set the filter for real or complex operation and to set the format for the transfer of weight data. The following tables give the meaning of the various control bits.

MODE WORD BIT DESCRIPTION

BIT	MEANING
0	Operating mode: 0 = complex 1 = real
1	Weight data transfer mode: 0 = all 32 bits are transferred 1 = only the 8 most significant bits are transferred
2–15	Unused

CONTROL WORD BIT DESCRIPTION

WORD 0, BIT	MEANING
0	Get a new X value
1	Get a new Y value; store the new error value
2	Update the weights
3	Cycle control 0 = proceed to the next control word 1 = return to the first control word
4	Reset the filter register before running; get a new error value
5	Output the filter value
6–11	X memory increment (0 is undefined)
12–15	Mu (gain) control 000 = 2^{-9} 017 = 2^{-24}

WORD 1, BIT	MEANING
0-11	First address in the weight memory
12-14	Unused
15	Output the error value

WORD 2, BIT	MEANING
0-11	Last address in the weight memory
12-15	Unused

DATA FORMATS

The input and output data transferred between the minicomputer and the LMS filter are packed into 16-bit data words. When the filter is operating in the complex mode, 8-bit data are always packed with the real part in the most significant bits and with the imaginary part in the least significant bits. This applies to X, Y, error, and filter output, as well as when the output transfer mode indicates that only 8-bit data should be transferred. When in the real operating mode, the filter will transfer two 8-bit data samples packed into a 16-bit word, with the first sample in the most significant bits and the second sample in the least significant bits. The weights are a special case when the filter is in the 32-bit transfer mode. It will take four 16-bit words to transfer each complex weight because each real weight and each imaginary weight contains 32 bits.

WORD 0 = most significant real part (bits 16-31)
WORD 1 = least significant real part (bits 0-15)
WORD 2 = most significant imaginary part (bits 16-31)
WORD 3 = least significant imaginary part (bits 0-15)

In the real operating mode, it will take only the first two of these words to transfer each real weight. The error memory is an exception to these rules: all sixteen complex words are always transferred regardless of the mode or configuration of the filter.

If the control data words are set to run more than one filter, the input data must be interleaved in the data buffers. The output data (from F or E) will also be interleaved.

ERRORS

As mentioned in the section on status and interrupts, the LMS filter will notify the minicomputer when a few simple errors are encountered. These include command errors, unexpected DMA cycle refused, and warnings of error or weight overflow. There are many other ways for the user to make an error in using the filter, and many of them can't be diagnosed by the hardware. Here are some illustrations of the kinds of errors which might occur:

Loading of the control memory from the filter DMA channel is normally terminated when a cycle bit is found in a control bit word. If the control buffer is out of sequence and the cycle bit is not found, the entire buffer could be emptied before an unexpected cycle refused occurs. Also, a data bit could be read as the cycle bit, and normal termination would leave incorrect data in the memory. If an attempt is made to load more than 32 control words, the filter will simply rewrite the first control words without informing the user of the error.

Loading or saving the X or W memories is normally terminated by a cycle refused on the X channel. If the DMA data buffers for this channel are circularly linked

(A linked to B, B linked to A), the operation will never terminate. (See section 5 on DMA channels.)

If either the configuration control words or a LOAD or SAVE instruction specifies locations beyond the maximum X or W address, the actual addresses used will start over at location zero. In calculating the maximum address, take into account the operating modes and the weight spacing on the X vector.

GENERAL EXAMPLE

This paragraph describes the programing sequence required to implement a typical filter configuration in the LMS filter hardware. The two general operations necessary are data buffer initialization and direct commands to the filter. Buffer initialization requires both data assembly and the loading of buffer addresses into the correct DMA channel control words. The following operations are necessary to load and start the filter:

STEP	COMMAND	DATA BUFFER INITIALIZATION
1	—	Configuration control (on filter channel)
2	Load Control	—
3	—	Initial W
4	—	*W initial address (on filter channel)
5	Load W	—
6	—	Initial X
7	—	*X initial address (on filter channel)
8	Load X	—
9	—	Next X, initial Y
10	Zero E	—
11	—	Error and filter output
12	Start	—

*These steps may have already been performed with step 1.

To stop the filter, either send the command STOP or allow a DMA buffer to empty. The data in the machine are saved by using the same sequence as above, where the commands are now SAVE W, X, and ERROR.

DISPLAYS

The LMS filter includes four output display channels intended for use with an oscilloscope. An 8-bit digital-to-analog converter provides the analog display for each output. Two of these display channels are fixed: both the real filter and the real error for the first filter in the control memory are always available. These outputs are on the front panel with the inputs for local running, but they are also valid when running from the computer.

The other two displays are selected from the front panel. Any of parameters X, F, Y, E, or W may be displayed. In addition, a test input is available as described later in this section, under TESTING. Either the real or the imaginary part of each input may be displayed. For handling multiple filters, the address of the control memory is also input from the front panel thumbwheels to decide when to sample the given parameter. The W display is an exception: all weights are always displayed.

LOCAL OPERATION

The LMS filter can be run as a stand-alone adaptive filter without the minicomputer, with the following restrictions: (1) only real filters can be run, (2) only one Y input is available, and (3) up to seven X input values can be used.

The control memory is loaded from the front panel and can be displayed for checking. The ZERO commands for the X, W, and E memories are also available. After the configuration control words have been set, the input sample period is chosen from the range of 8–1024 μs and the filter is started. A timing error indication is given if the sample period is set so short that the filter cannot finish a full cycle in that period. Additional inputs are available to zero the weights while running and to "freeze" (inhibit updating) the weights. The normal displays are available as just described. Note that the sample period can be set to any desired size by setting the panel RUN switch off and applying a pulse to the SINGLE CYCLE debugging input.

All front-panel switches are disabled by the panel switch. When the panel is off, only the minicomputer can control the hardware operation and data come only from the minicomputer. When the panel is on, all commands and data transfers from the minicomputer are ignored. In the local mode, it is important to check all test switches to make sure that they are in the correct position and not disabling some local function or data or providing other extraneous inputs.

TESTING

Because of the nature of an adaptive system, failures are much harder to identify and isolate. The LMS filter incorporates several features to make testing and debugging simpler and faster. As a result, the filter can be run without the minicomputer, as just described.

Furthermore, a front-panel section is provided solely for testing purposes. From here, repetitive pulses can be inserted to initiate all command functions which are normally provided by the minicomputer. The hardware may be switched to use external debug data for X, Y, W or E so that the data paths may be checked on a bit level. The relative X memory addressing may be disabled by a switch or a panel input. Finally, an external pulse may be provided for the internal logic term called RQUAL or Register Clock Qualifier. This input term is a single-cycle operation at the register level: one weight update and one filter update are done for each input pulse.

Internal connections for debugging are provided for the debug data input (described above), the multiplier outputs, the display D/A converter inputs and outputs, and output connections for the X and W memories. These connections are available internally on 16-pin IC sockets.

5 DIRECT MEMORY ACCESS (DMA)

The minicomputer interface (fig 8) provides a set of general-purpose input/output channels for convenient interfacing to a minicomputer. Eight CPU instruction channels and eight direct memory access (DMA) channels are provided. The present hardware has been tailored to the Interdata 7/16 machine, but the interface will work with any minicomputer with only minor changes. The following paragraphs describe the functions that are provided, the way to program the channels from the computer, and the interface terms.

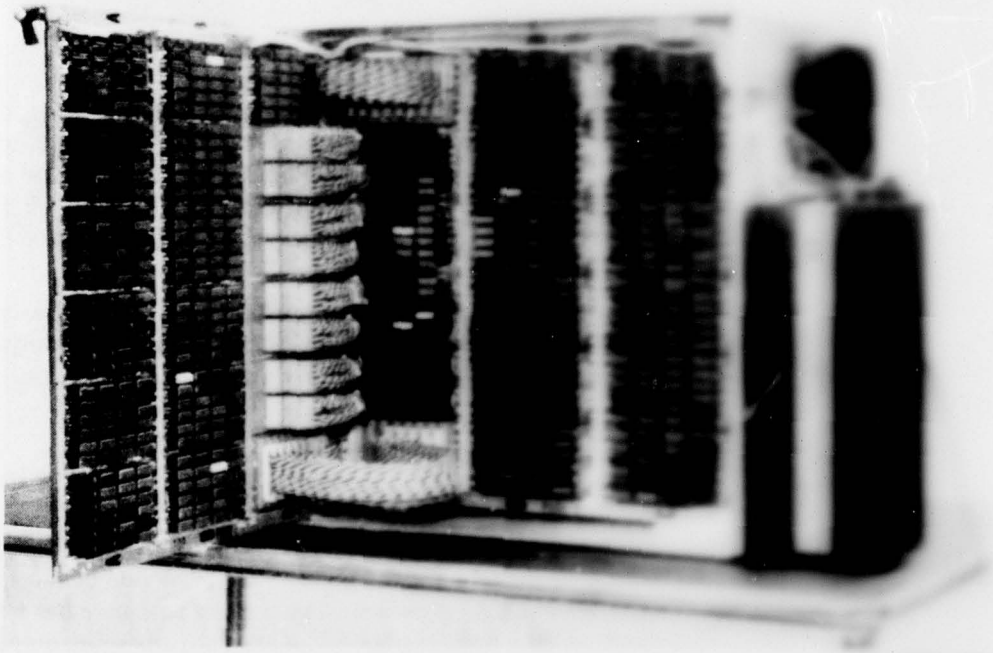


Figure 8. DMA hardware

GENERAL OPERATION

The operation of the I/O instruction channels needs little description; they are mainly buffers for the various CPU instructions dealing with input and output. The DMA channels are more complex in that they require some attention in handling and control. A total of 16 device addresses are used, each with an independent interrupt line. Both the 8 I/O instruction channel device numbers and the 8 DMA channel device numbers must be sequential. To simplify any changes, the initial device numbers are set into the DMA by positioning a series of switches. The DMA channels require an additional 8 bytes of reserved storage in memory for buffer pointers. These reserved locations are assumed to be sequential, with the first location identified to the hardware by a set of switches.

In all subsequent discussion, bit 0 is the least significant bit and bit 15 is the most significant bit.

PARAMETERS

The following table summarizes the parameters of the direct memory access hardware.

DIRECT MEMORY ACCESS HARDWARE PARAMETERS

Common Parameters

- General-purpose input/output channels
- 16 data bits in each channel
- Each channel own device address and interrupt line
- Channel device numbers are sequential
- Easily convertible to any minicomputer
- Software controlled
- Initial device numbers in hardware switches

I/O instruction channels

- Eight channels
- Deliver CPU input/output instructions
- Device commands
- Read status
- Each channel may interrupt at any time

DMA channels

- Eight channels
- High data rate
- Data buffering on input/output
- Channels run in parallel
- Buffers may be linked
- Programmable interrupts
- Address up to 20 bits

I/O INSTRUCTION CHANNELS

These eight channels provide an interface to the CPU input/output instructions. Since these instructions are generally slower than other instructions, their use should be restricted to short data transfers of no more than a few words. In addition to read and write instructions, there is an instruction for device commands and one to read the status of the device. The read/write instructions transfer 16 bits; the command and status instructions transfer only 8 bits. Each of these eight channels may interrupt the CPU at any time, and each is identified by its own device address.

DMA CHANNELS

The purpose of these channels is to provide a high data rate input/output capability for the minicomputer. Response time to device requests is decreased by a one-word data buffer on both input and output operations. The eight channels are independent and may be run in parallel. Data (16 bits) are read to or from sequential locations in the computer memory. The location of these data buffers is determined for each channel by a header block and by a reserved memory location containing the address of the header block. The data contained in the header are defined as follows:

WORD 0 = address of first word in data buffer

WORD 1 = address of last word in data buffer

WORD 2 = address of next header block

WORD 3 = link and interrupt flags and most significant address bits

The link and interrupt flags define the action taken by the DMA channel when the data buffer has been filled or emptied. If the link bit is set (bit 15), data transfers are continued by using a new buffer. This new buffer is described by a header block at the location in WORD 2. This linking may continue indefinitely, or a buffer or buffer chain may be linked in a loop. If the interrupt bit is set (bit 14), the DMA channel interrupts when the last data transfer for the current buffer is complete.

The DMA channels are designed to access data with up to 20 address bits. The most significant bits of the addresses in WORD 0 and WORD 1 are stored in WORD 3, bits 0–3 and 4–7 respectively. The header block locations are required to have the four most significant address bits equal to zero.

CONTROLLING THE DMA CHANNELS

The DMA channels are controlled by I/O instructions. Each channel has its own address and interrupt; in fact, each is treated as a separate device. These channels are controlled as follows.

COMMANDS

Direct control of the DMA channels is done with the command instruction, where bits 0 and 1 contain the function code describing the operation. There are four commands:

COMMAND BITS 1	COMMAND BITS 0	NAME	OPERATION
0	0	RESET	Reset all interface hardware regardless of which DMA channel was addressed.
0	1	STOP	Stop the addressed channel without completing any transfers which may be in progress.
1	0	START	Start the addressed channel. Command bit 2 indicates READ (=1) or WRITE (=0) operation.
1	1	Unused	No operation is performed.

Giving a START command while the channel is already operating is harmless, as the command will be ignored.

INTERRUPTS

When a data buffer transfer is complete, the DMA will interrupt the CPU if the interrupt bit is set (WORD 3, bit 14 of the associated header block). The timing of this interrupt is determined by two factors: whether a buffer link is to occur and whether an input or output operation is being performed. Output is treated differently because of the one-word output buffers: the interface hardware always stays one word ahead. The interrupt timing is summarized here.

(1) On input with no link	Interrupt when the last data word has been written into memory.
(2) On output with no link	Interrupt when the last data word has been taken from the output buffer by the device.
(3) On input or output, with link	Interrupt when the linking has been completed.

This timing definition assures the user that when an interrupt is received, all operations on the old data buffer have been completed.

STATUS

The current status of each channel can be determined using the Read Status instruction. The status word bit definitions follow the Interdata conventions and are as follows:

BIT	MEANING
0	Device unavailable
1	Unused
2	Unused
3	Device busy

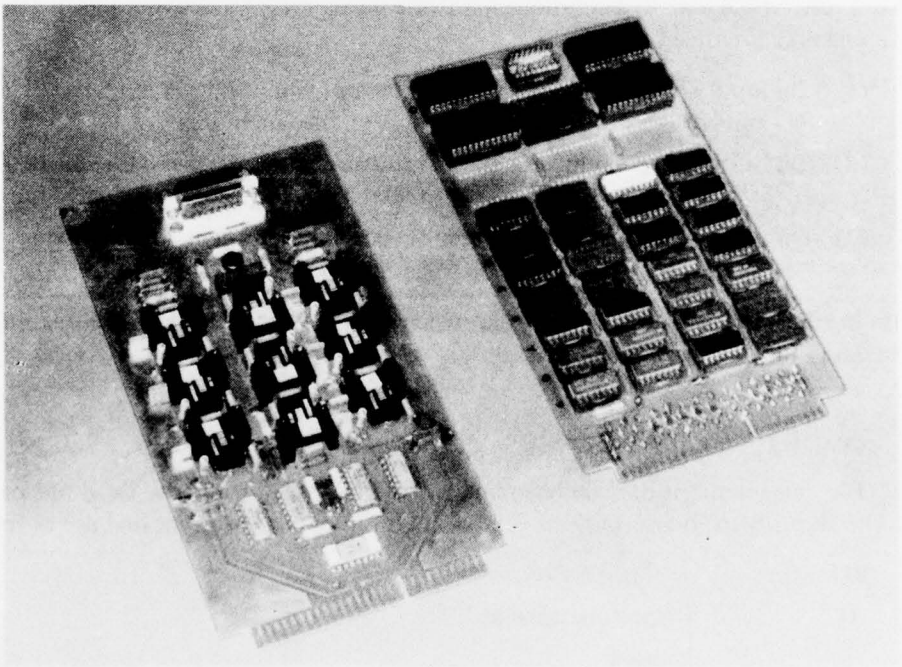
After a STOP command, the status word is set to indicate NOT BUSY. The status will also be set to NOT BUSY if the buffer is completed and no link is required. Bits 1 and 3 cannot both be true at the same time.

CURRENT BUFFER ADDRESS

As a data buffer is processed, the hardware keeps the memory address of the next word to be transferred. When the status is NOT BUSY, this word may be read using the READ instruction. When an input operation is being performed, this is the address into which the next device word will be written. On output, because of the one-word output buffers, the address read is one more than the address of the word in the output buffer.

6 RECEIVER FRONT END

The front end (see fig 9) contains the input processing for the channel adaptive receiver. The input channel (see fig 10) performs the proper filtering and down-mixing of the incoming signal to produce a baseband signal. The data out of the front end are transferred through the conversion equipment to the minicomputer, where they are stored until being sent to the adaptive filter for further processing. The exterior characteristics of the front end consist of one analog input and both a real and an imaginary analog output. The front end hardware is physically resident within the conversion equipment chassis (see section 3).



LRO 2433-4-76

Figure 9. Receiver Front End hardware.

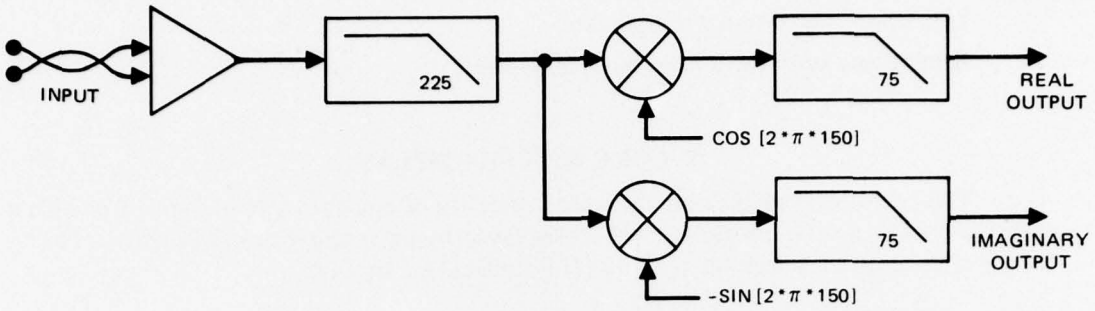


Figure 10. Front end hardware interconnections.

ANALOG HARDWARE

The input channel provides for a differential analog input with a maximum amplitude of ±5 V. The input signal is first low-pass filtered at 225 Hz (36 dB per octave) and then down-mixed to a pair of baseband components. This is done with two down-mixers with fixed local-oscillator frequencies of $\cos[2*\pi*150]$ and $-\sin[2*\pi*150]$ to produce both real and imaginary components. The signals are low-pass filtered at 75 Hz (36 dB per octave) to eliminate spurious frequency components before they are sent to the conversion equipment for data sampling. The components are then quantized and transferred to the minicomputer.

DIGITAL HARDWARE

The front end digital hardware performs two major functions: (1) It provides the real and imaginary local oscillator frequencies for the analog hardware. (2) It generates the external sample rate for the conversion equipment. The local oscillator signals, which are generated by means of a 16-bit frequency synthesizer, produce an average frequency accurate to within 0.004%. The conversion equipment external sample rate can be selected to be 200 Hz (fixed) or can be selected to come from an externally generated user signal. The generated sample rate, along with the sample rate divided by 2, 4, and 8, is made available to the user as an output for any synchronization that may be necessary.

PARAMETER SUMMARY

The important parameters of the receiver front end section are as follows:

Differential input signal, ±5 V max

Input signal bandwidth, 75 to 225 Hz

Complex down-mixer frequency, 150 Hz

Real and imaginary output components

Roll-off of filters, 36 dB per octave

Output signals, ±5 V max

Output signal bandwidth, ±75 Hz

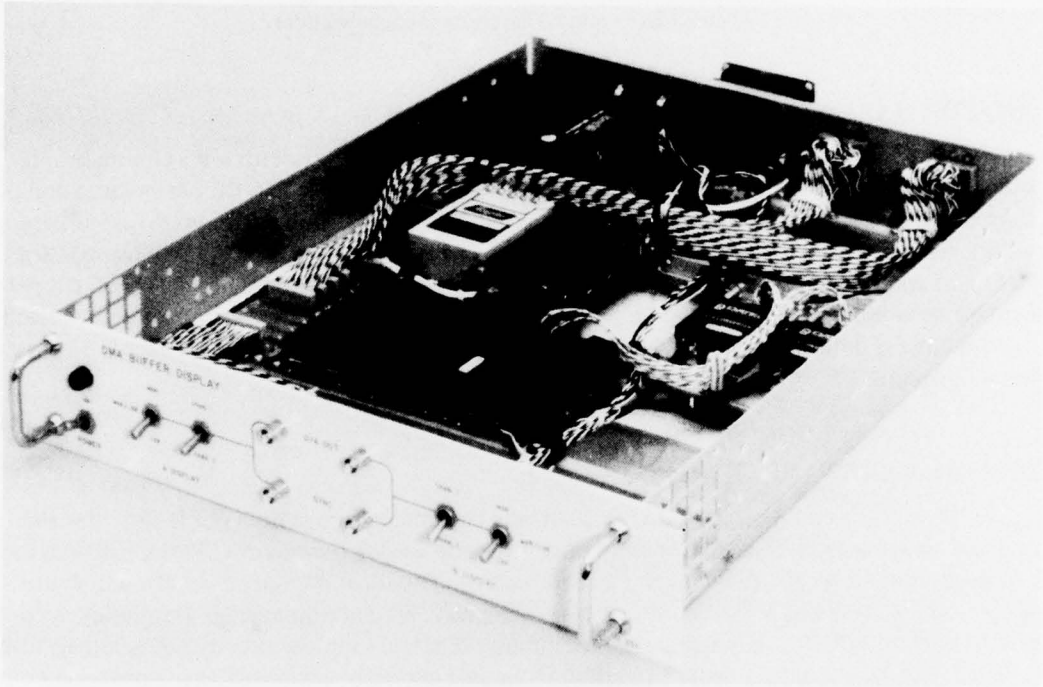
Internal A/D sample rate, 200 Hz

User-selectable external sample rate

Sample rate outputs divided by 1, 2, 4, and 8

7 CORE BUFFER DISPLAY

The core buffer display provides the capability of converting two channels of digital information into analog information to be displayed by an analyzer or oscilloscope. The displayed outputs are selectable from the front panel (see fig 11).



LHM 1412-3-75

Figure 11. Core buffer display hardware.

INPUT DATA INTERFACE

Two channels of interface are provided for input of data. This interface is directly compatible with the DMA hardware. An input channel basically consists of 16 data bits, a data request, request acknowledge, first word in buffer, and a data cycle refused. Requests are always made to the DMA channel and either can be controlled internally, at a 70-μs rate, or is user-selected externally. By the selection of an external sample rate, real-time outputs can be simulated or synchronized to a reference.

DATA OUTPUT

The data to be displayed are selected by the switches provided on the front panel along with an oscilloscope sync for each channel. The sync pulse occurs whenever the first word in buffer term is received. Each analog display channel can select from one of the two

input channels. In addition, the user can choose to display the most significant byte (MSB, 8-bits) only, the least significant byte (LSB, 8-bits), or both bytes sequentially (MSB first). The output amplitude is a maximum of 10 volts peak-to-peak. Whenever one of the channels runs out of data (receives a data cycle refused), output from the display goes to 0 volts.

PARAMETERS

The core buffer display parameters are summarized as follows:

Two DMA input channel interfaces

Each channel has 16 bits of data

Output channels and bytes selectable

Internal sample rate 70 μs

External user sample rate available

Outputs 10 volts peak-to-peak

Oscilloscope sync for each channel

8 OPERATING SYSTEM

To take best advantage of the capabilities of the adaptive filter system, it is necessary to program the Interdata 7/16. The method currently used for this system utilizes Concurrent Pascal, which is the minicomputer's operating system. The major components of it are described here along with information available to the user, such as various facts and techniques related to running Concurrent Pascal programs on the Interdata 7/16.*[17] The language is described by its author, Per Brinch Hansen,[18] among other documents.

GENERAL CODE COMPONENTS

There are three components of the run-time system in the Interdata. The first is the Kernel – a machine language program that implements a virtual computer with a set of instructions which are tailored to the running of concurrent processes. The Kernel also includes an interpreter for input/output instructions that reduces the machine and device dependency necessary in the Concurrent code. This IO Machine allows the device drivers to be written in Concurrent Pascal and keeps the details of interrupt code and device addressing out of the hands of the user. The IO Machine is an entry into the Kernel which provides a machine and device independent method of specifying a sequence of input or output operations with a peripheral device. We have defined a set of instructions which encompass the usual I/O functions: commands, status sensing, and reading or writing data. The actual method of data transfer to the device is hidden from the user, as are the details of interrupt handling.

---

*Additional information exists in an NOSC Fleet Engineering Department internal paper, Concurrent Pascal: User's Manual for the Interdata 7/16, by DM Cottel and JA Zaun, December 1976

[17]NOSC TD 146, Overview of Concurrent Pascal for the Interdata 7/16, by DM Cottel, 15 February 1978

[18]Concurrent Pascal Report, by PB Hansen, Information Science, California Institute of Technology, June 1975

The second kernel component is the run-time library for the compiler output. These assembly language routines implement functions which are often used by the compiler, such as routine entries, set operations, and the like. These are referenced at run-time to shorten the length of the output code.

The final run-time components are the utilities, device drivers, and the program you write in Concurrent Pascal. The utilities consist of various classes and monitors which are of general use in implementing concurrent programs. They facilitate the synchronizing of processes and the transfer of data and messages from one process to another.

The device drivers are a set of classes and monitors which automatically handle the interface to the hardware devices in the system. They transfer data from and to the various input/output devices attached to the Interdata 7/16 minicomputer. These system components handle all the details peculiar to each device, including the usually complex interrupt functions, and can control many input/output devices at the same time by overlapping their operations. These are written and need only be gathered up by the user when the current system requirements are known.

CONCURRENT PASCAL

The method of creating a Concurrent Pascal operating system is now briefly described. After deciding what is wanted from a system you may make a diagram of that system which reflects those requirements, showing what hardware is needed. At that point the device drivers required for that hardware are simply brought together. The handlers and utilities may be thought of and used exactly like independent building blocks. Just gather up the ones required for the system without worrying if they will fit together, because they do. As was pointed out earlier, all of the critical timing, interfacing, data spaces and transfers, and any other problems that can be associated with concurrent operating systems are very securely handled in the Kernel and IO Machine. The user need only be concerned with which pieces his system requires.

After the required pieces are located, the user then must inform the program how they are to be connected. Then the only thing left is to write the user program or process which controls the flow of data, and any processing. The compiler is excellent for finding errors prior to run-time; thus the debugging time of the program is greatly reduced.

COMPILER

We have modified the compiler to meet the needs of the Channel Adaptive Receiver program. As written by and obtained from Per Brinch Hansen, the compiled code ran on an interpreting kernel, whereas our compiler generates Interdata 7/16 code directly. As implemented on the Interdata 7/16, the compiler output consists of a beginning section which contains references to the library routines, the code itself, and a table of literal constants (strings, constant sets, and constant real numbers).

The language changes we have made have been restricted to changing the standard procedures and functions provided by the compiler. Some we added because they were needed by our run-time system; others were changed or eliminated because they didn't make sense in our system.

EXECUTION

Here is a brief description of what happens when you initially execute a Concurrent Pascal program with the Kernel in the Interdata. The Concurrent Program is, in effect, the operating system. It may be a simple program doing calculations, or it may be a complex system of concurrent processes such as a typical executive operating system for running user jobs. The initial system load is made from magnetic tape by using the Interdata relocatable loader.

After saving an image of the system on the disc, the Kernel initializes processes and peripherals such as the dispatch queue, the system clock, and the memory protection. At this point, the Initial Process is registered with the Kernel and execution begins normally. If at any time there are no processes executing, the initialization process resumes and sets the "wait" bit in the Program Status Word which is visible on the Interdata 7/16 front panel.

ERRORS

There are two sources of errors at run-time: those generated from the compiled code when an error state is detected and those not detectable from the compiled code but found and diagnosed by the Kernel. The error recovery in the Kernel is uniform for both sources of errors.

When an error occurs, the Kernel gets control. If the error was caused by a sequential program, the error number is saved with the calling process and execution of the sequential program is aborted. If the error happens in Concurrent code, the error is more serious and error recovery is probably not possible without booting the system. If there are no processes waiting to handle the error, the Kernel will halt, sending the message KERNEL HALT to the CRT and displaying the error number on the Interdata front panel. Core locations may be inspected with the front panel, or the system can simply be reinitialized from the disc by pressing the front panel INIT button.

REFERENCES

1. NOSC Technical Note NUC TN 1408, A Minicomputer Study, by CC Ross, August 1974. NOSC TNs are informal documents intended chiefly for internal use.
2. Interdata Inc User's Manual, Pub 29-261R03, July 1974.
3. Interdata Inc 16 Bit Series Reference Manual, Pub 29-398R01, August 1974.
4. Tektronix Inc 4023 Computer Display Terminal Users Instruction Manual, September 1974.
5. Wanco Inc Magnetic Tape Transport Operation and Maintenance Manual 201186, June 1973.
6. Diablo Systems Inc Series 40 Disk Drive Maintenance Manual, April 1974.
7. Computer Products Inc 8-Channel and 16-Channel High-Level Analog Input Module, PM021-096/097, January 1974.
8. Computer Products Inc Analog Output Modules, PM021-051C, August 1974.
9. Computer Products Inc Digital Input/Output and Analog Output Systems Command Equipment, PM070-004D, October 1974.

10. Computer Products Inc Conversion Equipment Controller Programming Specification, 02-292R01A22, July 1974.

11. Stanford University Stanford Electronics Laboratory Report SU-SEL-66-126, Adaptive Filters 1: Fundamentals, by B Widrow, December 1966.

12. Widrow, B, Adaptive Filters, Aspects of Network and System Theory, R Kalman and N DeClaris, ed, p 563-587, Holt, Rinehart, and Wilson, New York, 1971.

13. Adaptive Noise Cancelling: Principles and Applications, Proceedings of the IEEE, vol 63, no 12, December 1975.

14. NOSC Technical Note NUC TN 837, The Basic Principles of Adaptive Systems with Various Applications, by JM McCool, September 1972.

15. NOSC Technical Note NUC TN 1476, An Analysis of the LMS Adaptive Filter Used as a Spectral Line Enhancer, by JR Zeidler and DM Chabries, February 1975.

16. NOSC Technical Note NUC TN 1437, The Complex LMS Algorithm, B Widrow, JM McCool, and MS Ball, October 1974.

17. NOSC TD 146, Overview of Concurrent Pascal for the Interdata 7/16, by DM Cottel, 15 February 1978.

18. Concurrent Pascal Report, by PB Hansen, Information Science, California Institute of Technology, June 1975.